

# 成员变量和成员方法

- ◆ 成员变量和局部变量
- ◆ 成员变量和局部变量的区别
- ◆ 如何定义类的方法
- ◆ 方法的返回值
- ◆ 方法调用
- ◆ 为什么要用带参数的方法
- ◆ 如何使用带参数的方法
- ◆ 带两个参数的方法
- ◆ 对象数组类型的参数

# 成员变量和局部变量

- ▶ 变量声明的位置决定变量作用域
- ▶ 变量作用域确定可在程序中按变量名访问该变量的区域

...

```
for(int i = 0, a = 0; i < 4; i++){
```

```
    a++;
```

```
}
```

```
System.out.println ( a );
```

...

代码错误

a的作用域仅在for  
循环中

# 成员变量和局部变量

谁能使用这些变量？

Students类的方法，  
其他类的方法

方法1

方法2

成员变量

局部变量

局部变量

```
public class Students{
```

```
    变量1类型 变量1 ;  
    变量2类型 变量2 ;  
    变量3类型 变量3 ;
```

```
    public 返回类型 方法1(){  
        变量4类型 变量4;  
    }
```

```
    public 返回类型 方法2(){  
        变量5类型 变量5 ;  
    }
```

```
}
```



# 成员变量和局部变量的区别

## ▶ 作用域不同

- ▶ 局部变量的作用域仅限于定义它的方法
- ▶ 成员变量的作用域在整个类内部都是可见的

## ▶ 初始值不同

- ▶ Java会给成员变量一个初始值
- ▶ Java不会给局部变量赋予初始值

## ▶ 在同一个方法中，不允许有同名局部变量；

## ▶ 在不同的方法中，可以有同名局部变量

## ▶ 两类变量同名时，局部变量具有更高的优先级

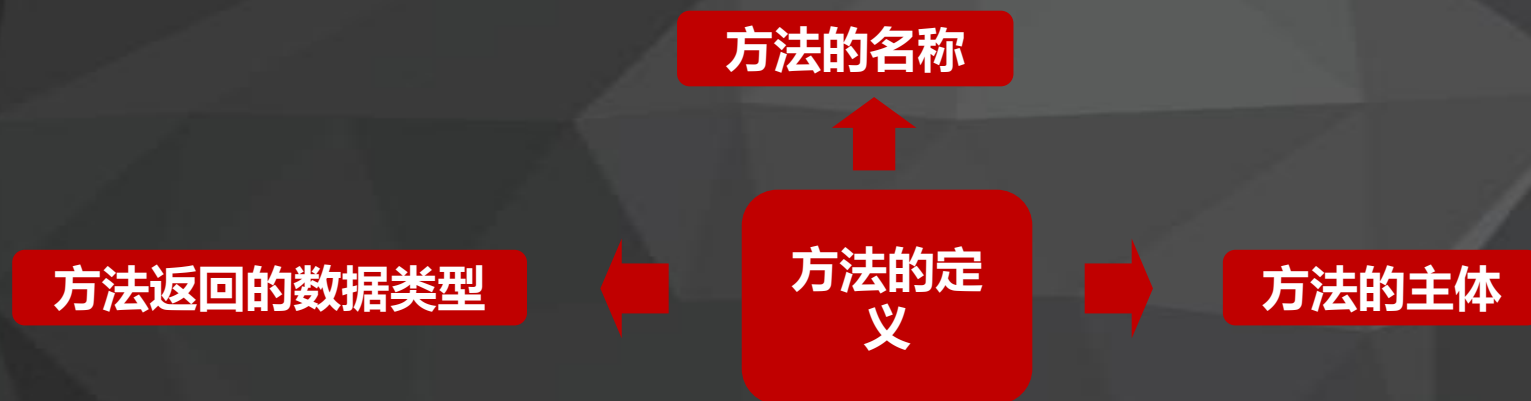
# 常见错误

```
public class Test {  
    int score1 = 88;  
    int score2 = 98;  
    public void calcAvg(){  
        int avg = (score1 + score2)/2;  
    }  
    public void showAvg(){  
        System.out.println("平均分是： " + avg);  
    }  
}
```

局部变量avg的作用域仅限于calcAvg()方法

# ◆ 如何定义类的方法

## ► 类的方法定义类的某种行为（或功能）



## 定义类的方法

步骤一：定义方法名以及返回值

```
public 返回值类型 方法名() {
```

```
//这里编写方法的主体
```

步骤二：编写方法体

```
}
```

# 方法的返回值

## 两种情况

- 如果方法具有返回值，方法中必须使用关键字 **return** 返回该值，返回类型为该返回值的类型

`return 表达式;`

作用：  
跳出方法  
给出结果

- 如果方法没有返回值，返回类型为 **void**

```
public class Student{  
    String name = "张三"  
    public void getName()  
        return name;  
}  
.....  
}
```

编译错误

返回类型要匹配



- ▶ 方法是个“黑匣子”，完成某个特定的应用程序功能，并返回结果
- ▶ 方法调用：执行方法中包含的语句

对象名.方法名();

- ▶ 方法之间允许相互调用，不需要知道方法的具体实现，提高了效率

情 况	举 例
Student类的方法a()调用Student类的方法b()，直接调用	<pre>public void a() {     b(); //调用b() }</pre>
Student类的方法a()调用Teacher类的方法c()，先创建类对象，然后使用 “.” 调用	<pre>public void a() {     Teacher t = new Teacher();     t.c(); //调用Teacher类的c() }</pre>

# ❖ 常见错误1

```
public class Student{  
    public void showInfo(){  
        return "我是一名学生";  
    }  
}
```

方法的返回类型为void，方法中不能有return返回值！

# ❖ 常见错误2

```
public class Student{  
    public double getInfo(){  
        double weight = 95.5;  
        double height = 1.69;  
        return weight, height;  
    }  
}
```

方法不能返回多个值！

# 常见错误3

```
public class Student{  
    public String showInfo(){  
        return "我是一名学生";  
    }  
    public double getInfo(){  
        double weight = 95.5;  
        double height = 1.69;  
        return weight;  
    }  
}
```



```
public class Student{  
    public String showInfo(){  
        return "我是一名学生";  
    }  
    public double getInfo(){  
        double weight = 95.5;  
        double height = 1.69;  
        return weight;  
    }  
}
```

多个方法不能相互嵌套定义！

# ❖ 常见错误4

```
public class Student{  
    int age=20;  
    if(age<20){  
        System.out.println("年龄不符合入学要求！");  
    }  
    public void showInfo(){  
        return "我是一名学生";  
    }  
}
```

**不能在方法外部直接写程序逻辑代码！**

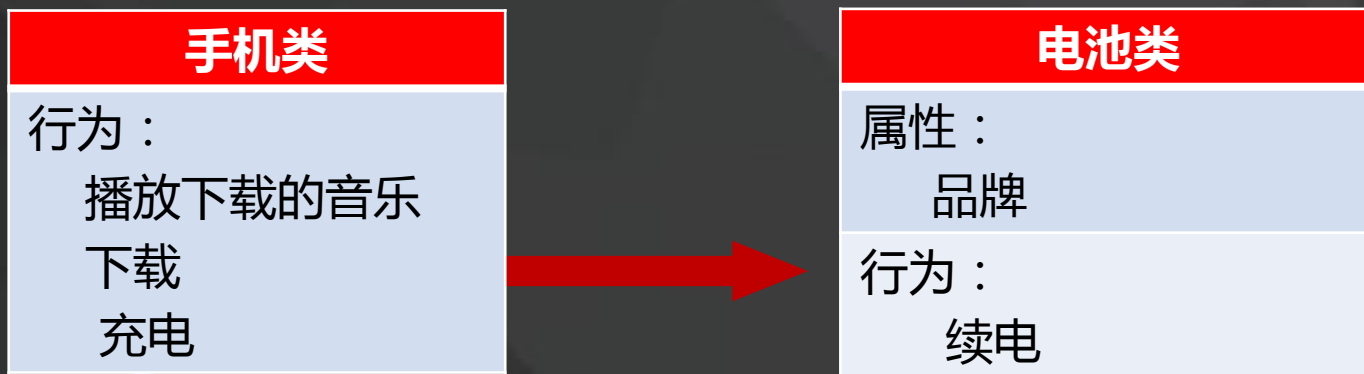
# 小结1

- ▶ 编写电池类（Cell）：具有品牌属性，可以续电
- ▶ 编写测试类（TestCell）

电池类
属性： 品牌
行为： 续电

## 小结2

- ▶ 编写手机类（Phone）：可以下载音乐，可以播放这些音乐，可以进行充电
- ▶ 重用电池类方法（Cell）
- ▶ 编写测试类（TestPhone）





# ◆ 为什么要用带参数的方法

## ► 工作原理

新鲜梨汁



# 如何使用带参数的方法

## 定义带参数的方法

```
public class Zhazhi{  
    public String zhazhi ( String fruit ) {  
        String juice = fruit + "汁";  
        return juice;  
    }  
}
```

参数列表：  
(数据类型 参数1，  
数据类型 参数2...)

## 调用带参数的方法

```
/*调用zhazhi方法*/  
Zhazhi myZhazhi = new Zhazhi();  
String myFruit = "苹果";  
String myJuice = myZhazhi.zhazhi(myFruit);  
System.out.println(myJuice);
```

调用方法，传递的参数要  
与参数列表一一对应

# ◆ 如何使用带参数的方法

语法：

该方法允许被访问调用的权限范围

传送给方法的形参列表

```
<访问修饰符> 返回类型 <方法名>(<形式参数列表>){  
    //方法的主  
}
```

方法返回值的类型

```
public class Stude
```

没有返回值

```
    String[] names = new String[30];  
    public void addName(String name){  
        //增加学生姓名  
    }  
    public void showNames (){  
        //显示全部学生姓名  
    }  
}
```

一个形式参数

# ◆ 如何使用带参数的方法

## ► 调用带参数的方法

实参列表

语法：对象名.方法名 ( 实参1, 实参2,..... , 实参n )

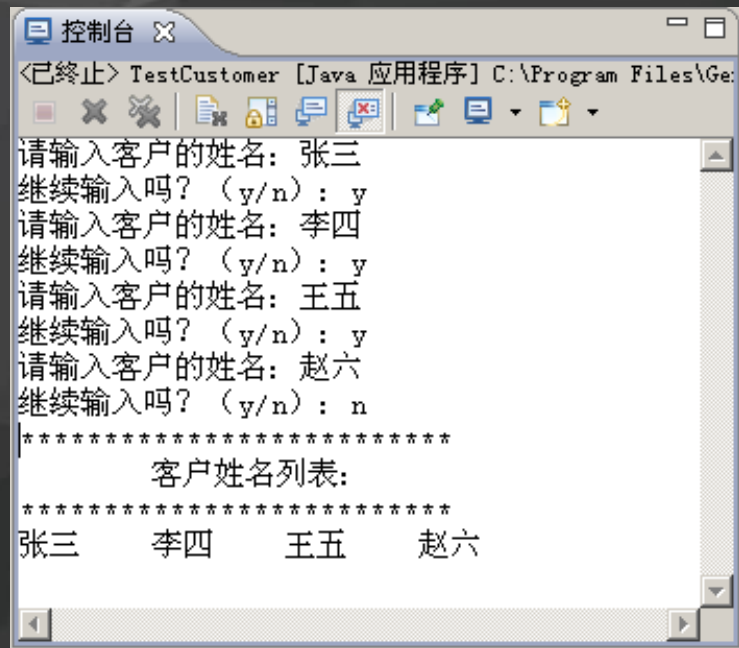
```
public static void main(String[] args) {  
    StudentsBiz st = new StudentsBiz();  
    Scanner input = new Scanner(System.in);  
    for(int i=0;i<5;i++){  
        System.out.print("请输入学生姓名 :");  
        String newName = input.next();  
        st.addName(newName);  
    }  
    st.showNames();  
}
```

先实例化对象，  
再使用方法

实参的类型、数量、顺序  
都要与形参一一对应

# ◆ 指导—实现客户姓名添加和显示

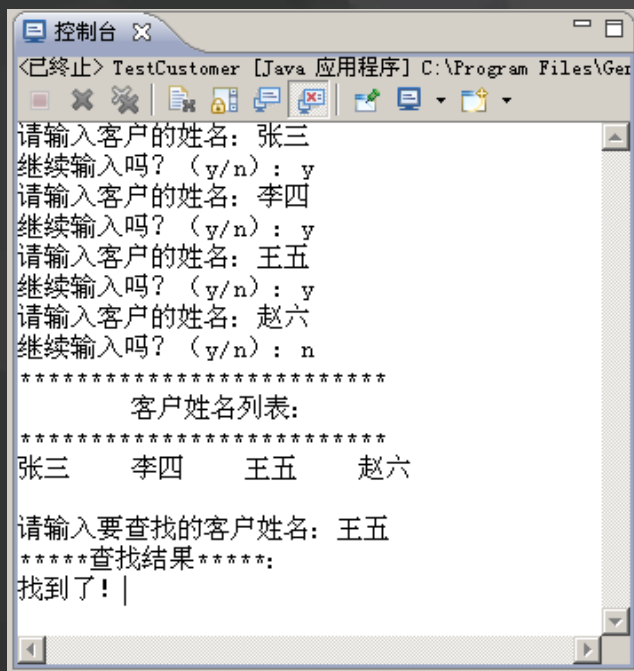
- ▶ 需求说明：
  - ▶ 创建客户业务类，实现客户姓名的添加和显示
- ▶ 实现思路：
  - ▶ 创建CustomerBiz类
  - ▶ 创建带参方法addName()
  - ▶ 创建方法showNames()
  - ▶ 创建测试类
- ▶ 难点指导：
  - ▶ 创建无返回值的带参方法



```
<已终止> TestCustomer [Java 应用程序] C:\Program Files\Ge...
请输入客户的姓名: 张三
继续输入吗? (y/n): y
请输入客户的姓名: 李四
继续输入吗? (y/n): y
请输入客户的姓名: 王五
继续输入吗? (y/n): y
请输入客户的姓名: 赵六
继续输入吗? (y/n): n
*****
          客户姓名列表:
*****
张三    李四    王五    赵六
```

# 练习——查找客户姓名

- ▶ 需求说明：
  - ▶ 根据需要，查找客户姓名，给出查找结果



```
<已终止> TestCustomer [Java 应用程序] C:\Program Files\Ger
请输入客户的姓名: 张三
继续输入吗? (y/n): y
请输入客户的姓名: 李四
继续输入吗? (y/n): y
请输入客户的姓名: 王五
继续输入吗? (y/n): y
请输入客户的姓名: 赵六
继续输入吗? (y/n): n
*****
      客户姓名列表:
*****
张三    李四    王五    赵六

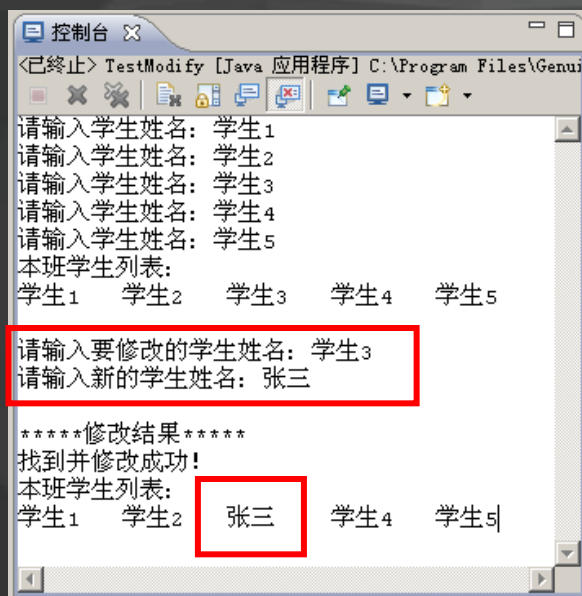
请输入要查找的客户姓名: 王五
*****查找结果*****:
找到了!|
```

提示：定义带参带返回值的方法  
`public boolean  
search(String name)`



# ◆ 带两个参数的方法

- ▶ 修改学生姓名，输入新、旧姓名，进行修改并显示是否修改成功



- ▶ 可以设计一个方法来实现，通过传递两个参数（需要修改的姓名、新姓名）来实现

# 带两个参数的方法

```
public class StudentsBiz {  
    String[] names = new String[30];  
    ...
```

返回值类型

```
public boolean editName (String oldName, String newName) {  
    boolean find = false; // 是否找到并修改成功  
    // 循环数组，找到姓名为oldName的元素  
    for(int i=0;i<names.length;i++){  
        if(names[i].equals(oldName)){  
            names[i] = newName;  
            find=true;  
            break;  
        }  
    }  
    return find;  
}
```

返回结果：  
boolean类型

```
public class TestModify {  
    public static void main(String[] args) {  
        ....
```

```
        System.out.print("\n请输入要修改的学生姓名：");  
        String oldname = input.next();  
        System.out.print("\n请输入新的学生姓名：");  
        String newname = input.next();  
        System.out.println("\n*****修改结果*****");  
        if( st.editName(oldname, newname) ){  
            System.out.println("找到并修改成功！");  
        }else{  
            System.out.println("没有找到，请重新输入！");  
        }  
        st.showNames();  
    }  
}
```

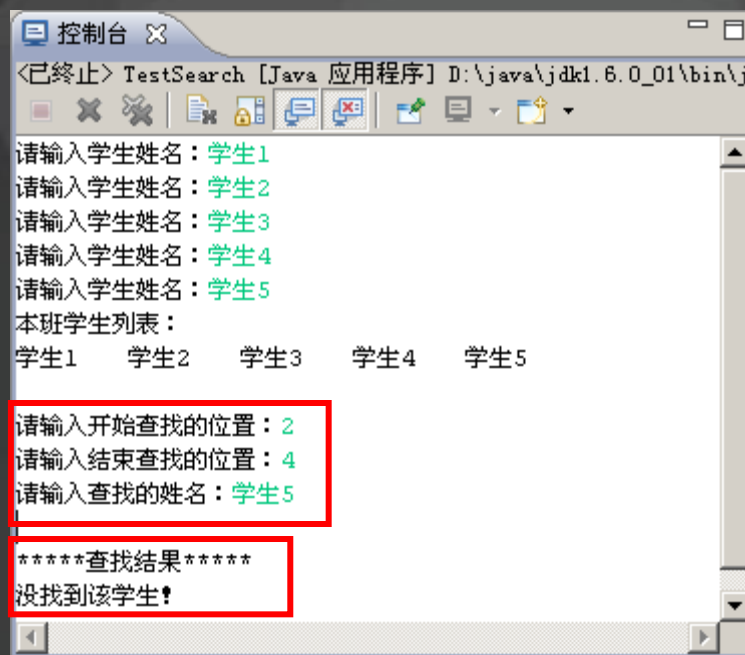
传递两个实参

根据返回值进行处理



# ◆ 带多个参数的方法

- ▶ 指定查找区间，查找学生姓名并显示是否修改成功



- ▶ 设计方法，通过传递三个参数（开始位置、结束位置，查找的姓名）来实现

# 带多个参数的方法

返回值类型

带有三个形参

```
public boolean searchName (int start,int end,String name){
```

```
    boolean find = false; // 是否找到标识
```

```
    // 指定区间数组中，查找姓名
```

```
    for(int i=start-1;i<end;i++){
```

```
        if(names[i].equals(name)){
```

```
            find=true;
```

```
            break;
```

```
        }
```

```
    }
```

```
    return find;
```

传递三个实参

```
        if(st.searchName(s,e,name)){
```

```
            System.out.println("找到了！");
```

```
        }else{
```

```
            System.out.println("没找到该学生！");
```

```
        }
```

返回结果  
boolean类型

# 常见错误1

//方法定义

```
public void addName(String name){
```

```
    //方法体
```

```
}
```

```
对象名.addName("张三");
```

//方法调用

```
对象名.addName(String "张三");
```

**调用方法时不能指定实参类型！**

# ◆ 常见错误2

//方法定义

```
public boolean searchName(int start, int end, String name){
```

//方法体

```
}
```

//方法调用

```
String s="开始";
```

```
int e=3;
```

```
String name="张三";
```

```
boolean flag=对象名.searchName(s,e,name);
```

形参和实参数据类型不一致！

# 常见错误3

//方法定义

```
public boolean searchName(int start,int end,String name){
```

//方法体

```
}
```

//方法调用

```
int s=1;
```

```
int e=3;
```

```
boolean flag= 对象名.searchName(s,e);
```

形参和实参数量不一致！

# 常见错误4

//方法定义

```
public boolean searchName(int start,int end,String name){
```

//方法体

```
}
```

//方法调用

```
int s=1;
```

```
int e=3;
```

```
String name="张三";
```

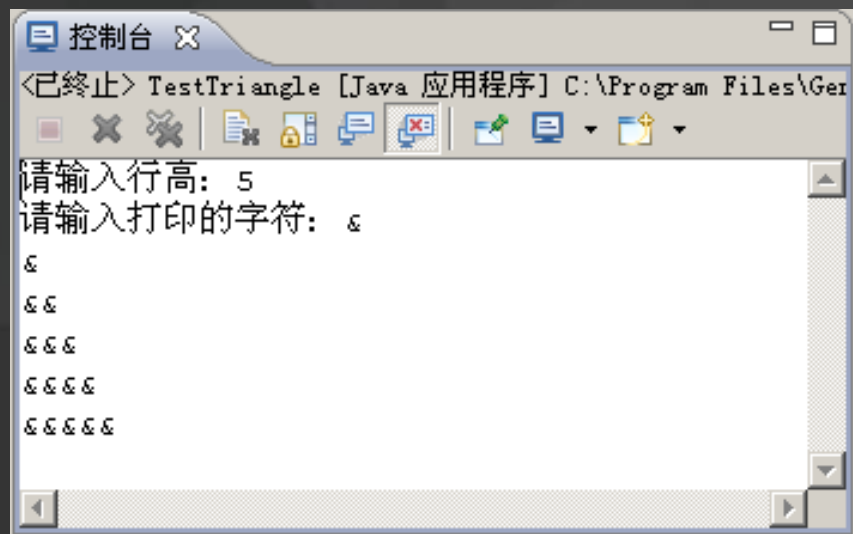
```
对象名.searchName(s,e,name);
```

调用方法后没有对返回值作任何处理！

# ◀ 练习——实现图形生成器

## ▶ 需求说明：

- ▶ 根据指定不同的行以及字符，生成不同的三角形



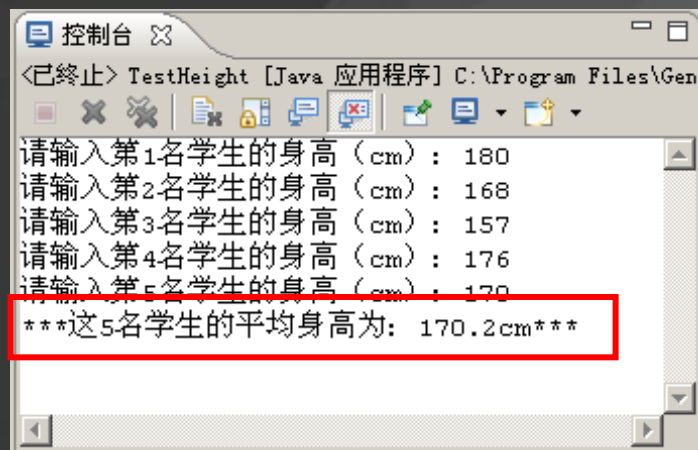
The screenshot shows a Java console window titled "控制台" (Console). The window displays the following text:

```
<已终止> TestTriangle [Java 应用程序] C:\Program Files\Ger  
请输入行高: 5  
请输入打印的字符: &  
&  
&&  
&&&  
&&&&  
&&&&&
```

The output shows a right-angled triangle of ampersands (&) with 5 rows. The first row has 1 character, the second has 2, the third has 3, the fourth has 4, and the fifth has 5.

# 对象数组类型的参数

## 计算学生的平均身高



```
控制台
<已终止> TestHeight [Java 应用程序] C:\Program Files\Gen
请输入第1名学生的身高 (cm): 180
请输入第2名学生的身高 (cm): 168
请输入第3名学生的身高 (cm): 157
请输入第4名学生的身高 (cm): 176
请输入第5名学生的身高 (cm): 170
***这5名学生的平均身高为: 170.2cm***
```

- ▶ Students类中定义身高属性
- ▶ Height类中定义方法，传递学生对象数组，求平均身高
- ▶ 测试类调用Height类的方法



# 对象数组类型的参数

```
public class Height {  
    public float getAvgHeight( Students[ ] stu){  
        float avgHeight=0;  
        float all=0;//所有学生的  
        int count=0;//学生计数  
        for(int i=0; i<stu.length; i++){  
            if(stu[i].height != 0){  
                all=all+stu[i].height;  
                count++;  
            }  
        }  
        avgHeight=all/count;  
        return avgHeight;  
    }  
}
```

```
public class TestHeight{  
    public static void main(String[ ] args) {  
        Students[ ] stu = new Students[5];  
        Height h=new Height();  
        ...  
        float avgheight=h.getAvgHeight(stu);  
        System.out.println("平均身高:"+avgheight+"cm");  
    }  
}
```

调用方法，传递对象数组

# 谢 谢

